How SkyWatch uses Serverless Cloud Computing to aggregate EO data for the masses

SKYWATCH

We believe that everyone on the planet should benefit from satellite data

Needs to be easy to obtain, easy to use, delivered in minutes, inexpensive

Problem

- Petabytes of data to process and distribute
- Pent up demand, bottlenecked by immature industry distribution vehicles and antiquated pricing structures

Goals

- no humans in the loop
- easy consumption of data by beginners and experts alike
- machine to machine focus
- low cost to purchase data
- low cost to operate
- timely processing and delivery (minutes not days)

Guiding Principles

100% Cloud based

- seamlessly scale as demand grows
- pay as you go
- storage is inexpensive
- devops simplified
  - take advantage of managed services

Microservices architecture

- app is broken up into independent services that have agreed to input and output interfaces eg. api or billing
- by contrast, monolithic architecture is where entire app sits on a server or cluster of servers (containers) as one entity
- microservices allow for:
  - use best tech for the job
  - easy to extend or upgrade
  - continuous integration and deploy

Serverless where possible

- management simplified
  - no servers to manage and scale
- stateless
  - the serverless function does not use state data to perform its task
  - totally reliant on inputs

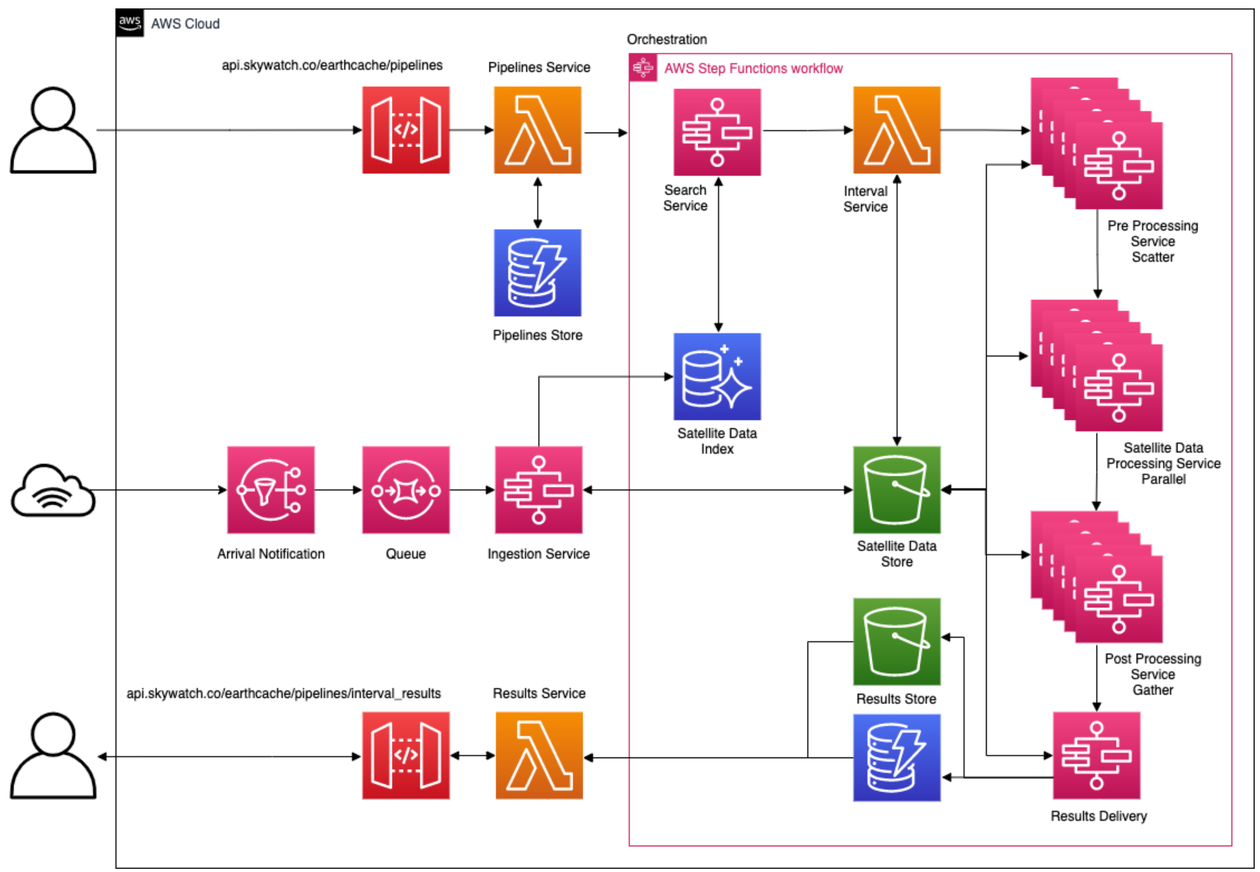
Solution - Cloud based serverless microservices architecture

Design patterns

- parallelism
  - stateless nature allows for multiple independent serverless microservice instances to be run at the same time
- scatter/gather
  - divide the problem domain into smaller pieces that can be worked on in parallel, and gather them together when processing is complete
- loosely coupled
  - where possible, one microservice should not be waiting for another to finish
- event driven
  - the invocation of a service should be initiated by an event that is monitored or subscribed to
  - can be managed by a queue

Implementation using Amazon Web Services (AWS)

- serverless
  - Lambda
  - Fargate
- orchestration
  - Step Functions
    - parallelism, scatter/gather
  - Simple Queue Service (SQS), Simple Notification Service (SNS), Kinesis
    - loose coupling, event driven
- storage
  - Simple Storage Service (S3)
    - key/value store
  - Relational Database Service (RDS)
    - Aurora PostgreSQL databases with PostGIS extension
  - DynamoDB
    - serverless noSQL database



Advantages

Scale when needed

- horizontal
  - run more instances
- vertical
  - more memory, power for serverless instances
  - seamless db or other infrastructure upgrades

Extend

eg. SAR or Hyperspectral

- easily add new or extend exisiting services

Lower unit costs

- savings passed on to customers

Lessons learned

- app-wide standards - paved road
- coordination of deploys that require many services
- be judicious about technical debt and cleanup

Contact:

SkyWatch Engineering Team

Roland Sing, Co-founder and Staff Platform Engineer

info@skywatch.com

skywatch.com